



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/822,468	04/12/2004	Mitchell Alsup	5500-92000	3134
53806	7590	06/07/2006	EXAMINER	
MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL (AMD)			ZALEPA, GEORGE D	
P.O. BOX 398			ART UNIT	
AUSTIN, TX 78767-0398			PAPER NUMBER	
			2183	

DATE MAILED: 06/07/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.

10/822,468

Applicant(s)

ALSUP ET AL.

Examiner

George D. Zalepa

Art Unit

2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 12 April 2004.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-26 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-26 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 12 April 2004 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. _____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date <u>6/29/05</u> . | 6) <input type="checkbox"/> Other: _____ |

Art Unit: 2183

DETAILED ACTION

1. The examiner has considered claims 1-26.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file:

- a. Declaration as filed on 12 April 2004.
- b. Information Disclosure Statement as filed on 29 June 2005.

Information Disclosure Statement

3. The references listed in the Information Disclosure Statement submitted on 29 June 2005 have been considered by the examiner (see attached PTO-1449).

Specification

4. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

The following title is suggested: Method and Apparatus for assigning an executable status to a trace cache entry dependent on a given branch prediction.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

6. Claims 1-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Witt et al. (US Pat. No. 6,256,729; herein referred to as "Witt") in view of Rotenberg et al ("Trace Cache: a

Art Unit: 2183

Low Latency Approach to High Bandwidth Instruction Fetching"; herein referred to as "Rotenberg".).

7. Regarding **independent claim 1**,

8. Ando discloses a method, comprising: predicting an execution path of a first conditional branch operation [see Witt, Col. 23, lines 55-60; Fig. 5, element 100]...in response to predicting said execution path, if a first operation...is not in said execution path according to said prediction, assign to said first operation a non-executable status indicative that said first operation is not in said execution path [see Witt, Col. 5, lines 5-11; Fig. 5, element 106; Examiner's note: It is clear that if a branch is predicted taken, instructions between the branch and it's target would not be in the execution path.]; detecting that said prediction is incorrect subsequent to assigning said non-executable status to said first operation [see Witt, Col. 15, lines 59-65; Examiner's note: Since Witt updates predictions after execution, it is clear that if an instruction is mispredicted, its prediction would be updated and thus on a subsequent execution of the branch instruction, those cancelled by a previous execution of the branch would be allowed to execute.]; assigning an executable status to said first operation in response to said detecting [see Witt, Col. 15, lines 59-65; Col. 2, lines 45-52; Examiner's note: Witt discloses selectively canceling instructions based upon a prediction, thus if an instruction is within the execution path (i.e., Fig. 8, element 12) it will be allowed to execute and thus have an executable status.], wherein said executable status is indicative that said first operation is in said execution path [see Witt, Fig. 8, element 12; Examiner's note: As stated previously, it is clear that if an instruction is not cancelled by Witt is in within the execution path of a branch instruction.].

9. Witt does not disclose the use of a trace cache.

10. Rotenberg does disclose the use of a trace cache [see Rotenberg, section 1.1, lines 8-11].

Art Unit: 2183

11. The advantage of utilizing a trace cache within the environment disclosed by Witt would have been to store frequently encountered instructions based on their execution rather than their static program order (as in an instruction cache), which would inherently increase the speed at which a processor operates as it would allow frequently executed groups of instructions to be accessed quicker [see Rotenberg, section 1.1, lines 1-6]. This advantage is desirable in the environment disclosed by Witt as the invention is intended to increase the performance of branch instructions and Rotenberg's trace cache is also intended to increase the speed at which branches can execute. Furthermore, trace caches would have been common knowledge at the time of invention and their effects on branch processing speed would have been motivation for one of ordinary skill in the art at the time of invention to utilize the trace cache instead of a standard instruction cache. Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to utilize a trace cache, as disclosed by Rotenberg, instead of the static instruction cache disclosed by Witt with the goal of increasing processor speed.

12. Regarding **claim 2**,

13. Witt and Rotenberg disclose the limitations as stated in **independent claim 1**.

14. Witt also discloses *preventing said first operation from executing in response to assigning said non-executable status to said first operation* [see Witt, Col. 5, lines 54-56].

15. Regarding **claim 3**,

16. Witt and Rotenberg disclose the limitations as stated in **independent claim 1**.

17. Witt also discloses *issuing said first operation from a scheduler for execution without refetching said first operation from said trace cache in response to assigning said executable status to said first operation* [see Witt, Col. 5, lines 23-31; Col. 9, lines 8-11; Examiner's note: Witt discloses in lines 23-31 maintaining target instructions in queue 20 with the goal of increasing fetch bandwidth.].

Art Unit: 2183

18. Regarding **claim 4**,

19. Witt and Rotenberg disclose the limitations as stated in **independent claim 1**.

20. Witt also discloses *determining a destination of said first operation in response to assigning said executable status to said first operation* [see Witt, Col. 6, lines 60-63; Examiner's note: x86 instructions inherently contain destinations.]; *determining that a second operation is dependent upon the destination of said first operation in response to determining said destination* [see Witt, Col. 10, lines 23-26]; and *configuring said second operation to receive a result from said first operation in response to determining that said second operation is dependent upon the destination of said operation* [see Witt, Col. 10, lines 23-26, Col. 10, line 64 to Coll. 11, line 8; Fig. 1, elements 24-28].

21. Regarding **claim 5**,

22. Witt and Rotenberg disclose the limitations as stated in **claim 5**.

23. Witt also discloses *storing in a destination list a respective destination specified by each unretired operation, wherein determining said destination of said first operation further comprises accessing said destination stored in said destination list* [see Witt, Fig. 1, element 28; Examiner's note: By definition, a reorder buffer contains this limitation.].

24. Regarding **claim 6**,

25. Witt and Rotenberg disclose the limitations as stated in **independent claim 1**.

26. Witt also discloses *in response to predicting said execution path, if said first operation stored in said entry...is in said execution path according to said prediction, assigning said executable status to said first operation* [see Witt, Col. 5, lines 5-11; Fig. 5, element 106; Examiner's note: It is clear that if a branch is predicted taken, instructions between the branch and its target would not be in the execution path and instructions after the target would be.]; *detecting that said prediction is incorrect subsequent to assigning said executable status to said first operation* [see Witt, Col. 15, lines 59-65; Examiner's note: Since Witt updates predictions after

Art Unit: 2183

execution, it is clear that if an instruction is mispredicted, its prediction would be updated and thus on a subsequent execution of the branch instruction, those allowed to execute by a previous execution of the branch would not be allowed to execute on the subsequent execution.]; and assigning said non-executable status to said first operation in response to said detecting [see Witt, Col. 15, lines 59-65; Col. 2, lines 45-52; Examiner's note: Witt discloses selectively canceling instructions based upon a prediction, thus if an instruction is not within the execution path (i.e., Fig. 8, element 11) it will not be allowed to execute and thus have a non-executable status.].

27. Regarding **claim 7**,

28. Witt and Rotenberg disclose the limitations as stated in **claim 6**.

29. Witt also discloses *determining a destination of said first operation in response to assigning said non-executable status to said first operation responsive to detecting that said prediction is incorrect* [see Witt, Col. 6, lines 60-63; Examiner's note: x86 instructions inherently contain destinations.]; *determining that a second operation is dependent upon the destination of said first operation in response to determining said destination* [see Witt, Col. 10, lines 23-26]; *and configuring said second operation to receive a result from a source other than said first operation in response to determining that said second operation is dependent upon the destination of said first operations* [see Witt, Fig. 1, element 24-26; Col. 10, lines 5-9]; .

30. Regarding **claim 8**,

31. Witt and Rotenberg disclose the limitations as stated in **independent claim 1**.

32. Witt also discloses *predicting an execution path of a second conditional branch operation stored in said entry* [see Witt, Fig. 9, element 110], wherein said first operation is dependent upon said first conditional branch operation and said second conditional branch operation [see Witt, Fig. 8, element 140], and wherein assigning an executable status to said first operation in response to said detecting that said prediction of said first conditional branch is

Art Unit: 2183

incorrect is dependent upon said first operation being in the predicted execution path of said second conditional branch operation [see Witt, Fig. 5, element 116; Examiner's note: It is clear that if a first prediction is incorrect ("No" stemming from element 100) an instruction's status is determined after the prediction of a second branch operation (element 112)].

33. Regarding **independent claim 9**,

34. Witt discloses a microprocessor comprising: branch prediction logic configured to predict an execution path of a first conditional branch operation stored in an entry of a ... cache [see Witt, Col. 23, lines 55-60; Fig. 4, element 60]; and dispatch logic coupled to said branch prediction logic and to said ... cache [see Witt, Fig. 4, element 68] and configured to: if a first operation stored in said entry of said ... cache is not in said execution path according to said prediction, assign to said first operation a non-executable status indicative that said first operation is not in said execution path [see Witt, Col. 5, lines 5-11; Fig. 5, element 106; Examiner's note: It is clear that if a branch is predicted taken, instructions between the branch and its target would not be in the execution path.]; detect that said prediction is incorrect subsequent to assigning said non-executable status to said first operation [see Witt, Col. 15, lines 59-65; Examiner's note: Since Witt updates predictions after execution, it is clear that if an instruction is mispredicted, its prediction would be updated and thus on a subsequent execution of the branch instruction, those cancelled by a previous execution of the branch would be allowed to execute.]; and assign an executable status to said first operation in response to said detecting [see Witt, Col. 15, lines 59-65; Col. 2, lines 45-52; Examiner's note: Witt discloses selectively canceling instructions based upon a prediction, thus if an instruction is within the execution path (i.e., Fig. 8, element 12) it will be allowed to execute and thus have an executable status.], wherein said executable status is indicative that said first operation is in said execution path [see Witt, Fig. 8, element 12; Examiner's note: As stated previously, it is clear that if an instruction is not cancelled by Witt is in within the execution path of a branch instruction.].

Art Unit: 2183

35. Witt does not disclose the use of a trace cache.

36. Rotenberg does disclose the use of a trace cache [see Rotenberg, section 1.1, lines 8-11].

37. The advantage of utilizing a trace cache within the environment disclosed by Witt would have been to store frequently encountered instructions based on their execution rather than their static program order (as in an instruction cache), which would inherently increase the speed at which a processor operates as it would allow frequently executed groups of instructions to be accessed quicker [see Rotenberg, section 1.1, lines 1-6]. This advantage is desirable in the environment disclosed by Witt as the invention is intended to increase the performance of branch instructions and Rotenberg's trace cache is also intended to increase the speed at which branches can execute. Furthermore, trace caches would have been common knowledge at the time of invention and their effects on branch processing speed would have been motivation for one of ordinary skill in the art at the time of invention to utilize the trace cache instead of a standard instruction cache. Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to utilize a trace cache, as disclosed by Rotenberg, instead of the static instruction cache disclosed by Witt with the goal of increasing processor speed.

38. Regarding **claim 10**,

39. Witt and Rotenberg disclose the limitations disclosed in **independent claim 9**.

40. Witt also discloses *a scheduler [see Witt, Fig. 1, element 20] coupled to receive said first operation from said dispatch logic [see Witt, Col. 9, lines 8-11] and configured to store an indication of said non-executable status of said first operation [see Witt, Col. 9, lines 53-56;* Examiner's note: Although not explicitly stated that the indication of validity is sent with the instruction, it would have been obvious that it is necessary for the correct functionality of the processor.].

Art Unit: 2183

41. Regarding **claim 11**,

42. Witt and Rotenberg disclose the limitations disclosed in **claim 10**.

43. Witt also discloses the *scheduler...further configured to prevent said first operation from executing in response to storing said indication of said non-executable status of said first operation* [see Witt, Col. 9, lines 53-56; Col. 9, lines 8-11; Examiner's note: Witt discloses sending instructions from fetch/scan unit (where instructions are marked as non-executable or executable) to a queue which stores them. There is no indication of removing this status nor is there any reasonable suggestion for it, therefore, it is clear that the status of the instruction is present in the queue.].

44. Regarding **claim 12**,

45. Witt and Rotenberg disclose the limitations disclosed in **claim 10**.

46. Witt also discloses the *scheduler...further configured to issue said first operation for execution without said dispatch logic refetching said first operation from said trace cache in response to said dispatch logic assigning said executable status to said first operation* [see Witt, Col. 5, lines 23-31; Col. 9, lines 8-11; Examiner's note: Witt discloses in lines 23-31 maintaining target instructions in queue 20 with the goal of increasing fetch bandwidth.].

47. Regarding **claim 13**,

48. Witt and Rotenberg disclose the limitations as stated in **independent claim 9**.

49. Witt also discloses *determining a destination of said first operation in response to assigning said executable status to said first operation* [see Witt, Col. 6, lines 60-63; Examiner's note: x86 instructions inherently contain destinations.]; *[determining] that a second operation is dependent upon the destination of said first operation in response to determining said destination* [see Witt, Col. 10, lines 23-26]; *[configuring] said second operation to receive a result from said first operation in response to determining that said second operation is dependent*

Art Unit: 2183

upon the destination of said first operation [see Witt, Col. 10, lines 23-26, Col. 10, line 64 to Coll. 11, line 8; Fig. 1, elements 24-28].

50. Regarding **claim 14**,

51. Witt and Rotenberg disclose the limitations as stated in **claim 13**.

52. Witt also discloses *dispatch logic...further configured to store a respective destination specified by each unretired operation in a destination list and to determine said destination of said first operation by accessing said destination stored in said destination list [see Witt, Fig. 1, element 28; Examiner's note: By definition, a reorder buffer contains this limitation.].*

53. Regarding **claim 15**,

54. Witt and Rotenberg disclose the limitations as stated in **independent claim 9**.

55. Witt also discloses *dispatch logic...further configured to: in response to predicting said execution path, if said first operation stored in said entry of said trace cache is in said execution path according to said prediction, assign said executable status to said first operation [see Witt, Col. 5, lines 5-11; Fig. 5, element 106; Examiner's note: It is clear that if a branch is predicted taken, instructions between the branch and it's target would not be in the execution path and instructions after the target would be.]; detect that said prediction is incorrect subsequent to assigning said executable status to said first operation [see Witt, Col. 15, lines 59-65; Examiner's note: Since Witt updates predictions after execution, it is clear that if an instruction is mispredicted, its prediction would be updated and thus on a subsequent execution of the branch instruction, those allowed to execute by a previous execution of the branch would not be allowed to execute on the subsequent execution.]; assign said non-executable status to said first operation in response to said detecting [see Witt, Col. 15, lines 59-65; Col. 2, lines 45-52; Examiner's note: Witt discloses selectively canceling instructions based upon a prediction, thus if an instruction is not within the execution path (i.e., Fig. 8, element 11) it will not be allowed to execute and thus have an non-executable status.].*

Art Unit: 2183

56. Regarding **claim 16**,

57. Witt and Rotenberg disclose the limitations as stated in **claim 15**.

58. Witt also discloses *dispatch logic further configured to: determine a destination of said first operation in response to assigning said non-executable status to said first operation responsible to detecting that said prediction is incorrect [see Witt, Col. 6, lines 60-63; Examiner's note: x86 instructions inherently contain destinations.]; determine that a second operation is dependent upon the destination of said first operation in response to determining said destination [see Witt, Col. 10, lines 23-26]; and configure said second operation to receive a result from a source other than said first operation in a response to determining that said second operation is dependent upon the destination of said first operation [see Witt, Fig. 1, element 24-26; Col. 10, lines 5-9].*

59. Regarding **claim 17**,

60. Witt and Rotenberg disclose the limitations as stated in **independent claim 9**.

61. Witt also discloses *predicting an execution path of a second conditional branch operation stored in said entry [see Witt, Fig. 9, element 110], wherein said dispatch logic is further configured to determine that said first operation is dependent upon said first conditional branch operation and said second conditional branch operation [see Witt, Fig. 8, element 140], and wherein said dispatch logic is further configured to assign an executable status to said first operation in response to said detecting that said prediction of said first conditional branch is incorrect is dependent upon said first operation being in the predicted execution path of said second conditional branch operation [see Witt, Fig. 5, element 116; Examiner's note: It is clear that if a first prediction is incorrect ("No" stemming from element 100) an instruction's status is determined after the prediction of a second branch operation (element 112)].*

62. Regarding **claims 18-26**,

Art Unit: 2183

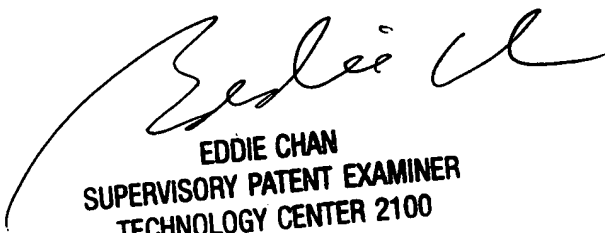
63. Claims 18-26 are rejected as being the apparatus disclosed in claim 9-17, respectively, with only the addition of a *system memory* [disclosed by Witt in Fig. 1, element 14].

Any inquiry concerning this communication or earlier communications from the examiner should be directed to George D. Zalepa whose telephone number is (571) 272-6754. The examiner can normally be reached on Monday-Friday (alt. Friday off).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie P. Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

George Zalepa
Examiner
Art Unit 2183
Randolph 2E74
Phone: (571)272-6754



EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100